IN THE SPECIFICATION:

Please replace paragraph [0043] with the following amended paragraph:

[0043] I/O Controller 240 includes a controller for PCI Bus 282 and may include controllers for System Management Bus (SMBus) 142, Universal Serial Bus (USB) 144, and the like. In an alternative embodiment, I/O Controller includes a controller for PCI Express bus. I/O Controller 240 also includes HOT Unit 250, effectively decoupling HOT Unit 250 from devices coupled to I/O Controller 240 via PCI Bus 282. Specifically, Hub-to-hub Interface 126 may be a high speed industry standard or proprietary bus coupling HOT Unit 250 to System Memory 130 via System Controller 120. Devices coupled to I/O Controller 240 share the bandwidth available on PCI Bus 282 which is typically lower than the bandwidth available on Hub-to-hub Interface 126. The location of HOT Unit 250 within I/O Controller 240 results in lower latency between HOT Unit 250 and both CPU 110 and System Memory 130 compared with latency between NIC 150 and CPU 110 shown in Fig. 1. Conventionally, low latency may be critical in communicating between a NIC, such as NIC 150 and an application program such as a software stack via a Driver [[255]]219. Low latency is particularly important for passing commands between NIC 150 and CPU 110, for example, to communicate that frame data stored in Driver Memory Space 135 is ready to be copied to Application Memory Space 125. Furthermore, because Hub-to-hub Interface 126 and Memory Bus 132 each support higher bandwidth than PCI Bus 282, HOT Unit 250 has higher bandwidth access to System Memory 130 than devices coupled to I/O Controller 240 via PCI Bus 282. Higher bandwidth access to System Memory 130 enables HOT Unit 250 to transfer received frames, sometimes referred to as "packets," to Application Memory Space 227 or Driver Memory Space 235 more quickly than a device coupled to I/O Controller 240 via a lower bandwidth bus such as PCI Bus 282.

Please replace paragraph [0054] with the following amended paragraph:

[0054] Fig. 4B is a flow diagram of method steps for receiving a frame, in accordance with one embodiment of the present invention. In step 424 HOT Unit 250 receives a frame via Input/Output Interface 242 and may partially process the frame producing a partially parsed frame and header data. In step [[416]]425 HOT Unit 250 determines if the frame was received on a delegated connection, and, if not, in step 440 HOT Unit 250 uploads the partially processed frame including its complete set of data link layer and network layer protocol header data to one or more legacy buffers. In step 442 TCP Stack 215 processes the partially processed frame uploaded to the legacy buffer.

Please replace paragraph [0055] with the following amended paragraph:

[0055] If, in step [[416]]425 HOT Unit 250 determines the frame was received on a delegated connection, then in step 426 HOT Unit 250 completes parsing of the frame, extracting the TCP payload data. In step 427 HOT Unit 250 determines if a user buffer is available, and, if so, then in step 428 HOT Unit 250 uploads the TCP payload data to one or more user buffers. If, in step 427 HOT Unit 250 determines a user buffer is not available, then in step 430 HOT Unit 250 uploads a portion of the payload data to a legacy buffer and notifies TCP Stack 215. In one embodiment the portion is specified by a "startup limit" value stored in the entry in the DCT 350 corresponding to the delegated connection. The "startup limit" is a variable that may take a maximum value equal to the maximum receive frame size and a minimum value as determined by Application Program 217 or TCP Stack 215.